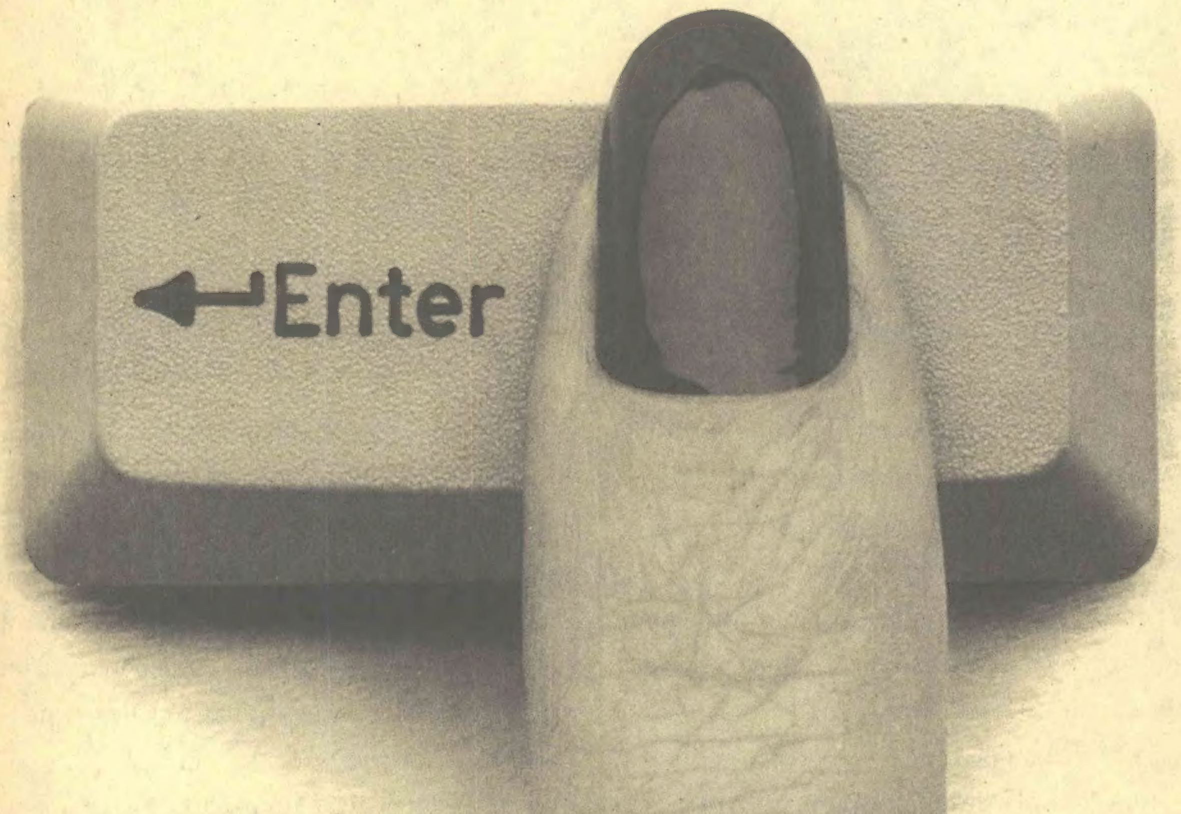
เพิ่มพลังให้คีย์บอร์ด



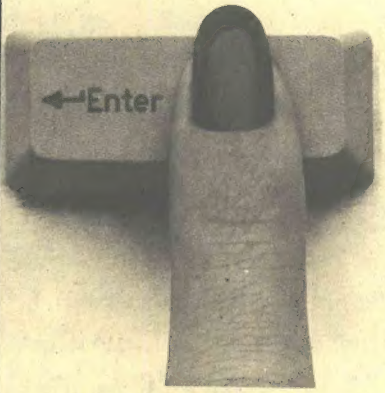
ขจร โรจนเมธินทร์

เกริ่นนำ

ถ้าคุณลองเปิดตารางรหัสแอสกีของ IBM ดูก็จะเห็นว่ามีค่าจาก 0-255 แต่ว่ารหัสของอักขระเหล่านี้คุณสามารถคีย์เข้าทางคีย์บอร์ดด้วยวิธีธรรมดาได้เพียงจากรหัส 32-126 เท่านั้น สำหรับรหัสที่เหลือซึ่งได้แก่ สัญลักษณ์กราฟิก สัญลักษณ์คณิตศาสตร์ และอักษรภาษากรีกนั้น คุณไม่สามารถคีย์เข้าทางคีย์บอร์ดตามธรรมดาได้ ยกเว้นจะใช้วิธีกดคีย์ Alt ค้างไว้พร้อมกับคีย์รหัสของอักขระที่ต้องการทาง Numeric keypad พร้อมกับปล่อยนิ้วที่กดคีย์ Alt ออก ซึ่งก็เป็นวิธีที่ยุ่งยากพอควรเพราะ คุณต้องจำรหัสของอักขระที่ต้องการได้ แล้วแถมยังต้องกดคีย์บอร์ดไม่ต่ำกว่า 4 ครั้งเพื่อให้ได้อักขระ 1 ตัว แต่ว่าโปรแกรม MBX.COM สามารถช่วยให้คุณคีย์อักขระเหล่านี้ได้

คีย์บอร์ดของ IBM

ในคีย์บอร์ดของ IBM จะใช้ไมโครโปรเซสเซอร์เบอร์ 8048 ทำหน้าที่คอยตรวจสอบว่ามีอะไรเกิดขึ้นกับคีย์บอร์ด



บ้าง จากนั้นก็จะรายงานไปให้ซีพียูรู้ผ่านทางฮาร์ดแวร์อินเตอร์รัพต์ เมื่อคุณกดนิ้วบนคีย์บอร์ดก็จะมีการนำเอา Make code ซึ่งก็คือ Scan code ของคีย์ที่ถูกกดไปใส่ไว้ใน Data port ของคีย์บอร์ด เช่น ถ้ากดตัว “A” ซึ่งมีค่า Scan code เท่ากับ 30 ค่า 30 นี้ก็จะถูกนำไปไว้ใน Data port เป็นต้น แต่พอคุณยกนิ้วจากคีย์ที่กำลังกดขึ้นคีย์บอร์ดก็จะส่งค่า Break code ซึ่งคือ ค่า Scan code บวก 128 ออกมาเพื่อทราบว่าคีย์อะไรที่เลิกกด จากนั้นก็จะมีการเรียกใช้อินเตอร์รัพต์เบอร์ 9 ที่อยู่ในรอมไบออส ให้อ่านค่าคีย์ที่ส่งมาทางพอร์ตเบอร์ 60h แล้วนำมาแปลความหมายเช่นถ้าเป็นอักขระธรรมดาก็จะเอา Scan code และ ASCII code ไปใส่ในคีย์บอร์ดบัฟเฟอร์ หรือถ้าเกิดมีการกด Ctrl-Break ก็จะต้องไปเรียกอินเตอร์รัพต์เบอร์ 1Bh ให้ทำงานเป็นต้น

คีย์บอร์ดบัฟเฟอร์

คีย์บอร์ดบัฟเฟอร์ก็คือบริเวณหน่วยความจำที่จองที่ไว้ในบริเวณซึ่งใช้เก็บข้อมูลของไบออส มีเนื้อที่ 32 ไบต์ สามารถเก็บอักขระได้ 16 ตัว โดยค่าในไบต์สูงจะเก็บค่า Scan code ส่วนในไบต์ต่ำจะเก็บค่ารหัสแอสกี ซึ่งถ้าอักขระใดไม่มีรหัสแอสกีค่าในไบต์ต่ำก็เป็น 0 จากนั้นโปรแกรมใดต้องการใช้ค่าที่รับจากคีย์บอร์ดก็สามารถเรียกใช้บริการจากอินเตอร์รัพต์เบอร์ 16h ได้

คีย์บอร์ดบัฟเฟอร์นี้มีลักษณะเป็น Circular queue โดยจะต้องมีบริเวณที่เก็บรายละเอียดของ คิว (Queue) ซึ่งใช้เก็บแอดเดรสที่ชี้จุดเริ่มต้นของบัฟเฟอร์ แอดเดรสที่ชี้จุดสิ้นสุดของบัฟเฟอร์ แอดเดรสที่ชี้หัวของคิว แอดเดรสที่ชี้หางของคิว เมื่อมีการอ่านค่าอักขระโดยใช้อินเตอร์รัพต์ 16h อักขระที่เก็บในบัฟเฟอร์ที่อยู่ต้นคิวก็จะถูกอ่านค่ามาจากนั้นก็จะต้องบวกค่าแอดเดรสที่ชี้ต้นคิวด้วย2 เพื่อให้ชี้ไปยังบัฟเฟอร์ที่เก็บอักขระตัวถัดไปที่จะถูกอ่าน หรือในกรณีที่อินเตอร์รัพต์ 9h นำเอาอักขระมาเก็บในบัฟเฟอร์บริเวณท้ายคิว จากนั้นแอดเดรสที่ชี้หางคิวซึ่งอยู่ก็ต้องถูกบวกด้วย 2 เช่นกันเพื่อปรับให้ค่าชี้ไปยังที่ว่างถัดไปที่จะใช้เก็บอักขระ ซึ่งเมื่อใดก็ตามที่ค่าของแอดเดรสที่ชี้หัวและหางคิวมีค่ามากกว่า ค่าแอดเดรสที่ชี้จุดสิ้นสุดของบัฟเฟอร์ก็จะถูกกำหนดให้ชี้ไปที่จุดเริ่มและจุดสิ้นสุดใหม่ เพราะฉะนั้นถ้าแอดเดรสของหัวคิวเท่ากับหางคิวก็แสดงว่าบัฟเฟอร์ว่างอยู่ หรือถ้าแอดเดรสที่ชี้หางคิวมากกว่าแอดเดรสที่ชี้หัวคิวอยู่สองก็แสดงว่าบัฟเฟอร์เต็มทำให้ไม่สามารถคีย์อักขระได้

นอกจากนี้ก็จะมีไบต์ที่ใช้แสดงสถานะว่ามีการกด Shift key ต่าง ๆ (Left Shift, Right Shift, Alt, Ctrl) ค้างอยู่หรือไม่และยังแสดงว่ามีการกด CapsLock, NumLock, ScrollLock ในสถานะ On หรือไม่

ดังนั้นถ้าเราสามารถเปลี่ยนค่าแอดเดรสที่ชี้จุดเริ่มต้นและแอดเดรสที่ชี้จุดสิ้นสุดของคีย์บอร์ดบัฟเฟอร์ให้ชี้ไปยังหน่วยความจำบริเวณอื่นที่ไม่ได้ใช้ เราก็สามารถเพิ่มขนาดของคีย์บอร์ดบัฟเฟอร์ให้ใหญ่ตามที่เราต้องการได้

การออกแบบโปรแกรม

1. โปรแกรม KBX นี้จะต้องเป็นโปรแกรมที่ฝังตัวบนหน่วยความจำ และทำหน้าที่เป็นคีย์บอร์ดไดรฟ์เวอร์ซึ่งจะคอยเปลี่ยนค่า Scan code ที่ส่งเข้ามาทางพอร์ต 60h ให้เป็นค่ารหัสแอสกีตามที่เรากำหนดให้กับคีย์บอร์ดใหม่

2. ต้องเลือกคีย์ที่จะเป็นสวิตช์ (Switch) เข้าสู่ตัวโปรแกรม เช่น ไซด์คิกใช้ Ctrl-Alt เข้าสู่โปรแกรม เป็นต้น ในโปรแกรม KBX นี้จะใช้คีย์ NumLock และคีย์ ScrollLock ที่จำเป็นต้องใช้ 2 คีย์ก็เนื่องจาก มีอักขระถึง 100 ตัว ที่ต้องกำหนดค่าให้คีย์บอร์ด จากรหัสแอสกี 127-178, 219-254, และ อักขระที่ใช้ตีกรอบอีก 22 ตัว จึงต้องแบ่งรหัสเป็น 2 ชุด คือสำหรับรหัส 127-168 และอักขระที่ใช้ตีกรอบแบบเส้นเดียวจะใช้ ScrollLock เป็นสวิตช์, รหัส 169-254 และอักขระที่ใช้ตีกรอบแบบเส้นคู่จะใช้ NumLock เป็นสวิตช์ แต่ถ้าต้องการใช้ NumLock, ScrollLock ในหน้าที่เดิมก็ต้องกด Shift + NumLock หรือ Shift + ScrollLock

3. ปกติการใช้คีย์บอร์ดขณะที่ไม่ได้เปิด CapsLock ถ้าต้องการให้อักขระที่คีย์เป็นตัวเราจะต้องกด Shift ค้างไว้เพื่อบอกให้ไบออสทราบว่าเราต้องการอักขระตัวใหญ่เช่นเดียวกันกับการใช้ KBX เมื่อต้องการคีย์อักขระที่กำหนดให้กับคีย์บอร์ดใหม่ก็ทำได้โดยกด NumLock หรือ ScrollLock ค้างไว้แล้วคีย์ตัวที่ต้องการเข้าไป

4. ต้องมี Window ที่จะแสดงรูปแบบที่กำหนดให้กับคีย์บอร์ดใหม่ให้ผู้

ใช้โปรแกรมเห็นเพื่อความสะดวก

การทำงานของโปรแกรม

โปรแกรม KBX สามารถแยกได้เป็นสองส่วนดังนี้

ส่วนแรก คือ INITIALIZE ทำหน้าที่ตรวจดูว่า KBX ถูกโหลดก่อนหน้าหรือยังถ้ายังก็จะโหลดโปรแกรมลงบนหน่วยความจำ จากนั้นจะตรวจสอบชนิดของจอว่าเป็นจอ EGA, CGA หรือโมโนโครม แล้วจึงกำหนดค่าตัวแปรต่าง ๆ ของโปรแกรม ส่วนต่อมาจะทำการเปลี่ยนแอดเดรสของอินเตอร์รัพต์ 9h ให้ชี้มาที่โปรแกรม KBX แทน

ส่วนที่สอง จะเป็นส่วนที่อยู่บนหน่วยความจำซึ่งจะคอยอ่านค่า Scan code จากพอร์ต 60h ถ้าค่าที่อ่านได้คือ 69 (Scan code ของ NumLock) แสดงว่ามีการกด NumLock และตรวจดูว่ามีการกด Ctrl, Alt ด้วยหรือไม่ ถ้ามีก็จะเรียก Int 9 เดิมเพื่อทำหน้าที่ของ Ctrl-NumLock แต่ถ้าไม่ได้กดก็ตรวจดูว่า Shift ซ้ายหรือขวาถูกกดหรือเปล่า ถ้ากดก็แสดงว่าต้องการใช้ในสภาพปกติ เช่นต้องการใช้คีย์ตัวเลขของ Numeric keypad จึงต้องไปเซตบิตที่ 5 ของ KB_STATUS ในไบออสให้เป็นหนึ่ง โดยการ XOR แล้วเรียก Int 9 เดิมแต่ถ้าไม่ได้กด Shift ก็แสดงว่าจะใช้คีย์ในชุดของ NumLock ก็จะเซตบิตที่ 1 ของ AUX_SHIFT แทน และขณะที่บิตที่ 1 ถูกเซตนี้ (NumLock กดค้างอยู่) ก็จะใช้ค่า Scan code เป็นอินเด็กซ์ (Index) เปิดตารางของ NumLock ด้วยคำสั่ง XLAT แล้วนำค่าไปใส่ในคีย์บอร์ดบัฟเฟอร์ และเมื่อใดที่จากพอร์ต 60h ได้ค่า 69 + 128 (เลิกกด NumLock) ก็จะต้องรีเซตบิตที่ 1 ของ AUX_SHIFT แล้วจะทำงานตามปกติ สำหรับกรณีของ ScrollLock ก็คล้ายกันขอให้ลองไล่โปรแกรมดู นอกจากนี้ถ้ามีการกด F10 ก็ต้องดูว่ามีการกด Alt ด้วยหรือไม่ ถ้ามีก็จะเรียก KB_DISPLAY ซึ่งเป็นรูทีนที่แสดงตำแหน่งของคีย์บอร์ด

ตารางที่ 1 แสดง Scan code ที่กำหนดรหัสแอสกีใหม่

Scan code เดิม	อักขระ	รหัสแอสกี	
		กด NumLock	กด ScrollLock
1	Esc	0	0
2 - 10	1 - 9	169 - 177	127 - 135
11 - 13	0, –, =	178, 219, 220	136, 137, 138
14 - 15	^H, ^I	0	0
16 - 21	Q W E R T Y	221 - 226	139 - 144
22 - 27	U I O P [ ]	227 - 232	145 - 150
28 - 29	^M ไม่มี	0	0
30 - 35	A S D F G H	233 - 238	151 - 156
36 - 41	J K L ; ' `	239 - 244	157 - 162
42 - 43	ไม่มี \	0	0
44 - 49	Z X C V B N	245 - 250	163 - 168
50 - 53	M , . /	251 - 254	0
54	ไม่มี	0	0
55 - 56	* ไม่มี	0	0
57	Space	0	0
58 - 70	ไม่มี	0	0
71 - 73	ไม่มี	218 194 191	201 203 187
74 - 76	ไม่มี	196 195 197	205 204 206
77 - 79	ไม่มี	180 179 192	185 186 200
80 - 83	ไม่มี	193 217 0 0	202 188 0 0

ในสภาพปกติ แต่ถ้ากด NumLock ค้างไว้ก็จะแสดงตำแหน่งของคีย์บอร์ดใหม่ที่เรากำหนดให้เมื่อกด NumLock เช่นเดียวกับกรณีที่กด ScrollLock และสามารถออกจากรูทีนนี้โดยการกด Esc

สำหรับ KBX นี้ถูกเขียนด้วยภาษาแอสเซมบลี ดังนั้นคุณจะต้องสร้างซอร์สโค้ดด้วยเอดิเตอร์ได้ KBX.ASM จากนั้นคอมพายล์โปรแกรมแล้วทำเป็น Binary file

```
MASM KBX;
LINK KBX;
EXE2BIN KBX KBX.COM
```

จากนั้นคุณก็จะได้ KBX. COM มาช่วยคุณทำงานตามความชอบใจ

รูปที่ 1 โปรแกรม KBX.ASM

```
;-----------------------------------------------------------------------------
;                    KBX.COM for the IBM Personal Computer - 1988
;-----------------------------------------------------------------------------
kb_data         equ  60h                        ;keyboard data port
kb_ctrl         equ  61h                        ;keyboard control port
eoi             equ  20h                        ;8259 END OF INTERRUPT value
int_ctrl        equ  20h                        ;8259 port address
;
;character for building window
;
upper_left      equ  201
upper_right     equ  187
lower_left      equ  200
lower_right     equ  188
vertical        equ  186
horizontal      equ  205
;
bios_data       segment at 40h                  ;BIOS data area
                org  17h
kb_status       db   ?                          ;keyboard status byte
                org  1Ah
buffer_head     dw   ?                          ;pointer to keyboard head
buffer_tail     dw   ?                          ;pointer to keyboard tail
                org  80h
buffer_start    dw   ?                          ;starting keyboard buffer address
buffer_end      dw   ?                          ;ending keyboard buffer address
bios_data       ends
;
code            segment para public 'code'      ;code segment
             assume    cs:code
             org       100h
begin:       jmp       initialize
;
copyright              db 'KBX Utility Copyright(C) 1988. SI 99;11',13,10,'$'
already_install        db 'KBX already installed.',13,10,'$'
enable_values          db 2Ch,28h,2Dh,29h,2Ah,2Eh,1Eh
screen_buffer          dw 1280 dup (?)          ;screen buffer
aux_shift              db 0                     ;auxiliary keyboard status byte
old_shift              db 0                     ;storage for old shift status
window_status          db 0                     ;window display status
adapter                db 2                     ;0 = MDA, 1 = CGA, 2 = EGA
video_segment          dw 0B800h                ;default video segment
border_attr            db 2Fh                   ;window border attribute
text_attr              db 1Bh                   ;window text attribute
cursor_mode            dw 0607h                 ;default cursor shape
cursor_position        dw ?                     ;saved cursor position
video_page             db ?                     ;saved video page number
video_offset           dw ?                     ;starting window offset address
```

▶

```
addr_6845       dw ?                    ;port address of CRTC
ibm             db 'IBM'                ;EGA ASCII signature
old9h           label dword
old9h_vector    dw 2 dup (?)            ;storage for interrupt 9 vector
;
fill_parms    dw 1,030Fh,12
              dw 2,050Fh,12
              dw 2,070Fh,12
              dw 2,090Fh,10
              dw 17,0537h,4
              dw 0,0737h,4
              dw 0,0937h,3
;
def_table     db 0,'1234567890-=',0
              db 0,'QWERTYUIOP[]',0
              db 0,'ASDFGHJKL;',39,96
              db 0,0,'ZXCVBNM,./'
              db 17 dup (0)
              db '789-456+123',0,0
;
scr_table     db 0,127,128,129,130,131,132,133,134,135,136,137,138,0
              db 0,139,140,141,142,143,144,145,146,147,148,149,150,0
              db 0,151,152,153,154,155,156,157,158,159,160,161,162
              db 0,0,163,164,165,166,167,168,0,0,0,0
              db 17 dup (?)
              db 201,203,187,205,204,206,185,186,200,202,188,0,0
;
num_table     db 0,169,170,171,172,173,174,175,176,177,178,219,220,0
              db 0,221,222,223,224,225,226,227,228,229,230,231,232,0
              db 0,233,234,235,236,237,238,239,240,241,242,243,244
              db 0,0,245,246,247,248,249,250,251,252,253,254
              db 17 dup (?)
              db 218,194,191,196,195,197,180,179,192,193,217,0,0
;
;------------------------------------------------------------------------
; Interrupt 9 handler. Execution comes here when a key is pressed or released.
;------------------------------------------------------------------------
new9h    proc       near
         sti                            ;set interrupt enable flag
         push       ax                  ;save AX
         in         al,kb_data          ;get scan code from keyboard
         cmp        al,69               ;NumLock key pressed?
         je         numlock_down        ;yes.
         cmp        al,69+128           ;NumLock key released?
         je         numlock_up          ;yes.
         cmp        al,70               ;ScrollLock key pressed?
         je         scrlock_down        ;yes.
         cmp        al,70+128           ;ScrollLock key released?
         je         scrlock_up          ;yes.
         cmp        al,68               ;F10 pressed?
         je         f10                 ;yes.
         cmp        aux_shift,0         ;NumLock or ScrollLock depressed?
         je         exit                ;no, then exit
         jmp        new_key             ;yes.
exit:    pop        ax                  ;restore AX
         jmp        old9h               ;goto old interrupt handler
;
numlock_down:
         call       get_status          ;get main shift status byte
         test       al,12               ;Ctrl or Alt pressed?
         jnz        exit                ;yes, goto normal handler
         push       ax                  ;save shift code
         call       reset_kb            ;reset the keyboard
         pop        ax                  ;restore shift code
         test       al,3                ;Shift key depressed?
         jnz        numlock1            ;yes.
         or         aux_shift,2         ;set NumLock shift bit
         jmp        end_int             ;exit
numlock1:
         push       ds                  ;save DS
         mov        ax,bios_data        ;point DS to BIOS data area
         mov        ds,ax
         assume     ds:bios_data
         xor        kb_status,32        ;toggle NumLock state
         pop        ds                  ;restore DS
         assume     ds:nothing
         jmp        end_int             ;exit
;
numlock_up:
         call       reset_kb            ;reset the keyboard
         and        aux_shift,253       ;clear NumLock shift bit
         jmp        end_int             ;exit
```

```
;
scrlock_down:
          call    get_status          ;get main shift status byte
          test    al,12               ;Ctrl or Alt pressed?
          jnz     exit                ;yes, goto normal handler
          push    ax                  ;save shift code
          call    reset_kb            ;reset the keyboard
          pop     ax                  ;restore shift code
          test    al,3                ;Shift key depressed?
          jnz     scrlock1            ;yes.
          or      aux_shift,1         ;set ScrollLock shift bit
          jmp     end_int             ;exit
scrlock1:
          push    ds                  ;save DS
          mov     ax,bios_data        ;point DS to BIOS  data area
          mov     ds,ax
          assume  ds:bios_data
          xor     kb_status,16        ;toggle ScrollLock state
          pop     ds                  ;restore DS
          assume  ds:nothing
          jmp     end_int             ;exit
;
scrlock_up:
          call    reset_kb            ;reset the keyboard
          and     aux_shift,254       ;clear ScrollLock shift bit
;
end_int:  mov     al,eoi              ;issue EOI to 8259 controller
          out     int_ctrl,al
          pop     ax                  ;clean up the stack
          iret                        ;retrun form interrupt
;
f10:
          call    get_status          ;get current shift status
          test    al,8                ;Alt key pressed?
          jz      exit                ;no, goto normal handler
          call    reset_kb            ;yes, reset keyboard
          mov     al,eoi              ;issue EOI value
          out     int_ctrl,al
          cmp     window_status,0     ;is the window already up?
          jne     f10_1               ;yes, then ignore keypress
          push    bx                  ;save BX
          call    kb_display          ;pop up keyboard display
          pop     bx                  ;restore BX
f10_1:    pop     ax                  ;clean up the stack
          iret                        ;exit
;
new_key:  push    ax                  ;save scan code
          call    reset_kb            ;reset keyboard
          pop     ax                  ;recover scan code
          test    al,80h              ;is the high bit set?
          jnz     newkey3             ;yes, then don't process it
          push    bx                  ;save BX
          lea     bx,num_table        ;point BX to NumLock key table
          cmp     aux_shift,1         ;ScrollLock depressed?
          jne     newkey1             ;no, then continue
          lea     bx,scr_table        ;yes, then adjust BX
newkey1:  mov     ah,al               ;transfer scan code to AH
          dec     al                  ;set AL relative to zero base
          xlat    num_table           ;get ASCII code from table
          or      al,al               ;is it zero?
          je      newkey2             ;yes, then don't process this key
          call    insert_char         ;insert character into kb buffer
newkey2:  pop     bx                  ;restore BX
newkey3:  mov     al,eoi              ;issue EOI
          out     int_ctrl,al
          pop     ax                  ;restore AX
          iret
new9h     endp
;
;------------------------------------------------------------------------
; RESET_KB resets the keyboard
;------------------------------------------------------------------------
reset_kb proc     near
          in      al,kb_ctrl          ;get control port value
          mov     ah,al               ;save it
          or      al,80h              ;set high bit of control value
          out     kb_ctrl,al          ;send reset value
          mov     al,ah               ;get original control value
          out     kb_ctrl,al          ;enable keyboard
          ret
reset_kb endp
;
```

```
;-----------------------------------------------------------------------------
: GET_STATUS returns the main keyboard shift status byte in AL
;-----------------------------------------------------------------------------
get_status          proc      near
        push    ds                              ;save DS
        mov     ax,bios_data                    ;point DS to BIOS data area
        mov     ds,ax
        assume  ds:bios_data
        mov     al,kb_status                    ;get status byte in AL
        pop     ds
        assume  ds:nothing
        ret
get_status          endp
;
;-----------------------------------------------------------------------------
; INSERT_CHAR inserts the character code in AX into the keyboard buffer.
: Entry:      AH,AL - scan code, ASCII code
;-----------------------------------------------------------------------------
insert_char         proc      near
        push    dx
        push    ds
        mov     bx,bios_data                    ;point DS to BIOS data area
        mov     ds,bx
        assume  ds:bios_data
        mov     bx,buffer_tail                  ;get current buffer tail address
        mov     dx,bx                           ;transfer to DX
        add     dx,2                            ;calculate next buffer position
        cmp     dx,buffer_end                   ;did we overshoot the end?
        jne     insert1                         ;no, continue
        mov     dx,buffer_start                 ;yes, then wrap around
insert1: cmp    dx,buffer_head                  ;is the buffer full?
        je      insert2                         ;yes.
        mov     [bx],ax                         ;deposit character into buffer
        mov     bx,dx                           ;advance buffer tail
        mov     buffer_tail,bx                  ;record it new value
insert2: pop    ds
        assume  ds:nothing
        pop     dx
        ret
insert_char         endp
```

```
;
;-----------------------------------------------------------------------------
; KB_DISPLAY open a window showing all possible key definition
;-----------------------------------------------------------------------------
kb_display          proc      near
;
;Make sure current video mode is an 80-column text mode.
;
        mov     ah,15                           ;get current video mode
        int     10h
        cmp     al,2                            ;video mode 2?
        je      kb1                             ;yes, continue
        cmp     al,3                            ;video mode 3?
        je      kb1                             ;yes, continue
        cmp     al,7                            ;monochrome ?
        je      kb1
        ret                                     ;unsupported mode terminate
kb1:    mov     window_status,1                 ;set status flag
        push    cx
        push    dx
        push    si
        push    di
        push    ds
        push    es
        push    cs
        pop     ds
        assume  ds:code
        push    cs
        pop     es
;
;Save page number, cursor mode, and cursor position. then hide cursor.
;
        mov     video_page,bh                   ;store video page number
        mov     ah,3                            ;get current cursor mode
        int     10h
        mov     cursor_mode,cx                  ;store cursor mode
        call    read_cursor                     ;get cursor position
        mov     cursor_position,ax              ;save position
        mov     ah,1                            ;hide cursor
        mov     ch,20h
        int     10h
;
;Save the portion of video memory that will be overwritten.
;
        cmp     adapter,1               ;disable video if this is a CGA
        jne     kb2
        call    disable_cga
kb2:    call    save_screen             ;copy video memory into buffer
;
;Open the keyboard display window.
;
        call    open_window             ;open the window
        cmp     adapter,1               ;enable CGA video
        jne     kb3
        call    enable_cga
kb3:    lea     si,def_table            ;point SI to default key table
        call    fill_window             ;draw unshifted key definitions
;
kb4:    mov     ah,1                    ;check buffer for character
        int     16h
        jz      kb5                     ;goto if buffer is empty
        mov     ah,0                    ;get the character
        int     16h
        cmp     al,27                   ;is it the ESC key?
        je      kb7                     ;yes, close window and exit
kb5:    mov     al,aux_shift            ;get auxilary shift byte
        cmp     al,2                    ;is it = 2?
        jna     kb6                     ;yes.
        mov     al,2                    ;no, set it to 2
kb6:    cmp     al,old_shift            ;has the shift status changed?
        je      kb4                     ;no, loop
        mov     old_shift,al            ;record current shift status
        mov     bl,83                   ;number of keys on the keyboard
        mul     bl
        mov     si,ax                   ;transfer to SI
        add     si,offset def_table     ;complete offset address in SI
        call    fill_window             ;write key equivalents to window
        jmp     kb4                     ;go back for more
;
;The ESC key was pressed. Close the window and exit
;
kb7:    cmp     adapter,1               ;blank CGA video
        jne     kb8
        call    disable_cga
kb8:    call    restore_screen          ;restore contents of video memory
        cmp     adapter,1               ;enable CGA video
        jne     kb9
        call    enable_cga
kb9:    mov     ah,2                    ;restore cursor position
        mov     bh,video_page
        mov     dx,cursor_position
        int     10h
        mov     ah,1                    ;restore cursor mode
        mov     cx,cursor_mode
        int     10h
        mov     window_status,0         ;reset status flag
        pop     es
        pop     ds
        pop     di
        pop     si
        pop     dx
        pop     cx
        ret
kb_display      endp
;
;-----------------------------------------------------------------------
; SAVE_SCREEN saves the block of video memory that will be overwritten.
;-----------------------------------------------------------------------
save_screen     proc    near
        mov     si,182                  ;set zero page window offset in SI
        mov     cl,video_page           ;get video page in CX
        xor     ch,ch                   ;clear CH
        jcxz    save2                   ;branch if page zero
save1:  add     si,1000h                ;add one page length
        loop    save1                   ;loop until address is correct
save2:  mov     video_offset,si         ;save starting window address
        push    ds                      ;save DS
        mov     ds,video_segment        ;point DS to video memory
        assume  ds:nothing
        lea     di,screen_buffer        ;point ES:DI to screen buffer
        mov     cx,11                   ;11 lines to save
save3:  push    cx                      ;save line counter
        mov     cx,58                   ;58 character per line
        cld                             ;clear DF
        rep     movsw                   ;transfer one line to buffer
        pop     cx
        add     si,44                   ;point SI to next line
        loop    save3                   ;loop until all lines are saved
        pop     ds                      ;restore DS
        assume  ds:code
        ret
save_screen     endp
;
;------------------------------------------------------------------
; RESTORE_SCREEN restores saved video memory.
;------------------------------------------------------------------
restore_screen  proc    near
        mov     es,video_segment        ;point ES to video segment
        mov     di,video_offset         ;point DI to window area
        lea     si,screen_buffer        ;point DS:SI to screen buffer
        mov     cx,11                   ;11 lines to restore
restore1:
        push    cx                      ;save line counter
        mov     cx,58                   ;58 character per line
        rep     movsw                   ;transfer one line to buffer
        pop     cx
        add     di,44                   ;point SI to next line
        loop    restore1                ;loop until all lines are saved
        ret
restore_screen  endp
;
;------------------------------------------------------------------
; DISABLE_CGA control CGA video output.
;------------------------------------------------------------------
disable_cga     proc    near
        mov     dx,3DAh                 ;status register port address
disable1:
        in      al,dx                   ;read status
        test    al,8                    ;vertical retrace active?
        jz      disable1                ;no, then wait until it is
        sub     dx,2                    ;point DX to MSR
        mov     al,25h                  ;load disable value
        out     dx,al                   ;disable video
        ret
disable_cga     endp
;
;------------------------------------------------------------------
; ENABLE_CGA control CGA video output.
;------------------------------------------------------------------
enable_cga      proc    near
        mov     ah,15                   ;get current video mode
        int     10h
        lea     bx,enable_values        ;point BX to table of values
        xlat    enable_values           ;get value to enable signal
        mov     dx,3D8h                 ;MSR address
        out     dx,al                   ;enable video
        ret
enable_cga      endp
;
;------------------------------------------------------------------
; READ_CURSOR reads the cursor position.
; Exit:         AH, AL - row, column
;------------------------------------------------------------------
read_cursor     proc    near
        push    bx
        push    cx
        push    dx
        mov     ah,3                    ;read cursor position
        int     10h
        mov     ax,dx                   ;return value by AX
        pop     dx
        pop     cx
        pop     bx
        ret
read_cursor     endp
;
;------------------------------------------------------------------
; OPEN_WINDOW writes the new window to video memory.
;------------------------------------------------------------------
open_window     proc    near
```

```
        mov     es,video_segment        ;point ES to video segment
        mov     di,video_offset         ;point DI to start of window
        mov     al,upper_left           ;get first character code
        mov     ah,border_attr          ;and first attribute byte
        stosw                           ;write
        mov     cx,56                   ;do the next 56 characters
        mov     al,horizontal
        rep     stosw
        mov     al,upper_right          ;finish the first line
        stosw
        add     di,44                   ;advance DI to next line
        mov     cx,9                    ;9 identical lines next
open1:  push    cx                      ;save line counter
        mov     al,vertical             ;do first character
        push    ax                      ;save character/attribute
        stosw
        mov     cx,56                   ;do the next 56 character
        mov     al,32
        mov     ah,text_attr
        rep     stosw
        pop     ax                      ;restore character/attribute
        stosw                           ;finish the line
        add     di,44                   ;advance DI to next line
        pop     cx                      ;retrieve line count
        loop    open1                   ;loop until all 9 are done
        mov     al,lower_left           ;first character on last line
        stosw
        mov     cx,56
        mov     al,horizontal
        rep     stosw
        mov     al,lower_right
        stosw
        ret
open_window     endp
;
;-----------------------------------------------------------------------
; FILL_WINDOW writes a set of key definitions to the open window.
; Entry:        DS:DI - key definition table address
;-----------------------------------------------------------------------
fill_window     proc    near
        mov     bh,video_page           ;retrieve video page number
        lea     di,fill_parms           ;point DI to parameter table
        mov     cx,7                    ;7 lines to write
fill1:  push    cx                      ;save counter
        add     si,word ptr [di]        ;adjust table next index
        mov     dx,[di+2]               ;set starting cursor position
        mov     cx,[di+4]               ;set number of character
        call    writeln                 ;write one line
        add     di,6                    ;advance parameter table index
        pop     cx                      ;retrieve count
        loop    fill1                   ;loop until done
        ret
fill_window     endp
;
;-----------------------------------------------------------------------
; WRITELN writes one line of key equivalents to the open window.
; Entry:  DS:SI   - character string address
;         BH      - video page
;         CX      - number of character
;         DH,DL   - starting row, column
;-----------------------------------------------------------------------
writeln proc    near
        push    cx                      ;save counter
        mov     ah,2                    ;position the cursor
        int     10h
        lodsb                           ;get one character
        mov     ah,10                   ;print it
        mov     cx,1
        int     10h
        add     dl,3                    ;advance cursor position
        pop     cx                      ;retrieve counter
        loop    writeln
        ret
writeln endp
;
;-----------------------------------------------------------------------
; INITIALIZE prepares the body of the program for residency.
;-----------------------------------------------------------------------
initialize      proc    near
;
```

```
;Check if KBX already installed.
;
        mov     ax,3509h                ;get address of int 9h
        int     21h
        cmp     bx,offset new9h         ;did KBX already install?
        jne     init1                   ;no, then load KBX
        lea     dx,already_install      ;write already install
        mov     ah,9                    ;write function
        int     21h
        int     20h                     ;terminal program
;
;See if the display adapter is an EGA.
;
init1:
        mov     ax,0C000h               ;set ES to EGA BIOS segment
        mov     es,ax
        mov     di,1Eh                  ;point DI to signature location
        lea     si,ibm                  ;point SI to 'IBM' text
        mov     cx,3                    ;three bytes to compare
        cld                             ;clear DF
        repe    cmpsb                   ;check three bytes
        je      init2                   ;branch if signature found
;
;Determine whether adapter is a CGA or MDA.
;
        dec     adapter                 ;decrement assume value
        mov     ah,15                   ;get current video mode
        int     10h
        cmp     al,7                    ;is it mode 7?
        jne     init2                   ;no, it's a CGA
;
;The display adapter is an MDA.
;
        dec     adapter                 ;set adpater value for MDA
        sub     video_segment,800h      ;modify video segment value
        mov     border_attr,70h         ;modify border attribute
        mov     text_attr,0Fh           ;modify text attribute
        mov     cursor_mode,0C0Dh       ;modify cursor shape
;
init2:  mov     cx,cursor_mode          ;set scan lines in CX
```

```
        mov     ah,1                    ;set cursor
        int     10h
;
        mov     ax,40h                  ;point ES to BIOS data area
        mov     es,ax
        mov     di,63h                  ;point DI to address word
        mov     dx,es:[di]              ;get CRTC address
        mov     addr_6845,dx            ;save it
;
        lea     dx,copyright            ;write copyright message
        mov     ah,9                    ;write string DOS function
        int     21h
        mov     ax,3509h                ;get address of interrupt 9 vector
        int     21h
        mov     old9h_vector,bx         ;save it
        mov     old9h_vector[2],es
        mov     ah,25h                  ;set new address of interrput 9
        lea     dx,new9h
        int     21h
;
        lea     dx,initialize           ;set address for resident
        int     27h                     ;terminate but stay resident
initialize      endp

code    ends
        end     begin
```